הסבר עבור הקוד בקבצים ArchProj.c, ArchProj.h

בקבצים אלה נמצאים כל הפונקציות, המבנים והמשתנים שנחוצים על מנת לתפעל את המעבדים והצנרת.

מבני נתונים:

IF\_ID: החוצץ בין F ל-D. מחזיק את קידוד ההוראה, ה-PC ודגל של VALID

ID\_EX: החוץ בין D ל-E. מחזיק את ההוראה המפוענת וביטי הבקרה.

EX\_MEM: החוצץ בין E ל-M. מחזיק את ההוראה המפוענחת, סיגנלי הבקרה ובקשות לזיכרון

MEM\_WB: החוצץ בין M ל-W. מחזיק את ההוראה המפוענחת, סיגנלי הבקרה ובקשות לזיכרון

STATE: מצב הליבה. מחזיק את ה-PC הנוכחי, מצב הרגיסטרים ומספר ההוראה בכל שלב בצנרת

משתנים גלובליים חשובים:

Files: קבצי הקלט והפלט

Fnames: שמות דיפולטיביים לקבצים

R: שומר את ערכי הרגיסטרים של כל ליבה

Count: סופר את מספר המחזור לכל ליבה

Insts: מספק ההוראות שבוצעו בליבה

Decode\_stall: מספר ה-stall שנוזר מהאזרדים

Mem\_stall: מספר ה-stall בגלל החטאות מטמון

Halt\_prop: מעקב אחר התקדמות HALT בליבה לצורך עצירת ריצה

Branch\_taken: מעקב אחר פקיצות שנלקחו

Hazard: מעקב אחר האזרדים בליבה

State: מצב הליבה

Ifid: הרגיסטרים בחוצץ בין F ו-D

Idex: הרגיסטרים בחוצץ בין D ו-E

Exmem: הרגיסטרים בחוצץ בין E ו-M

Memwb: הרגיסטרים בחוצץ בין M ו-'

פונקציות:

Init: מאתחל את הצנרת

Load\_instruction\_memory: טעינת זיכרון הנתונים לליבה

Load\_main\_memory: טעינת הזיכרון הראשי

Fetch: ביצוע שלב ה-F של ההוראה הבאה/ הקפיצה שנלקחה

Decode: ביצוע שלב ה-D להוראה שנכנסה

Execute: ביצוע שלב ה-E וספירת סך הפקודות שהתבצעו

Memory: ביצוע שלב ה-M טיפול בload/store-

Writeback: ביצוע שלב ה-W וכתיבה לרגיסטרים באיחור של שעון אחד.

Hazard\_detector: בחינת מצב הצנרת של ליבה מסוימת ואיתור האזרדים מסוג WAW ובין D ל-E או D ל-W.

Core\_stopped: בדיקה האם ליבה סיימה את הריצה שלה.

Trace: ייצוא טרייס של הליבה לקובץ

Regout: יצוא מצב הרגיסטרים בסוף הריצה לקובץ

Stats: יצוא סטטיסטיקות ריצה לקובץ

Main: סימלוץ ריצת המעבד:

טוענים את הזיכרון הראשי והזיכרונות לכל הליבות ומאתחלים את מבני הנתונים. מאפסים את עצירת הליבות, stallים והאזרדים. בלולאה while (שמתסיימת כאשר כל הליבות סיימו את ריצתן) בודקים עבור כל ליבה האם סיימה לרוץ. אם לא, רצים על הצנרת בסדר הפוך (מ-WB ל-F) כדי לקרוא את הערכים הנכונים. במידה ויש cachestall (החטאה ב-cache) מחכים למידע מהמטמטון ומעכבים את הצנרת עבור ליבה זו (לא מבצעים F עד M). אם אין cachestall אז ממשיכים לMEM. במידה ויש HAZARD אז מקפיאים את ריצת הצנרת (מ-E ל-F). אחרת מבצעים אותם.

לאחר ביצוע כל שלבי הצנרת הנדרשים בודקים hazard למחזור הבא ומדפיסים פלט עבור המחזור הנוכחי. אם הפקודה היא HALT אז מתחילים קאונטר שמתזמן את עצירת הליבה ומרימים דגל לכל הפקודות שנכנסות שהן invalid. קוראים ל-bus\_Step שיסמלץ את ריצת הבאס והמטמונים.

הסבר עבור הקוד בקבצים **Bus.c** , **bus.h**:

בקבצים אלה נמצאים כל הפונקציות, המבנים והמשתנים שנחוצים על מנת לתפעל את מערכת הזיכרון.

**Init\_caches**: מאתחלת את המבנים של המטמונים.

**Free\_caches**: משחררת את הזכרון של המטמונים.

**DSRAM**: מבנה עבור DSRAM של ליבה כלשהי.

**TSRAM**: מבנה עבור TSRAM של ליבה כלשהי.

**PR\_REQ**: מבנה עבור בקשת מעבד.

**BUS\_REQ**: מבנה עבור בקשת BUS.

כאשר לליבה מסוימת יש פקודה שניגשת לזכרון, היא בונה מבנה מסוג PR\_REQ שבו יש את כל הפרמטרים לבקשה, ומשתמשת בפונקציות **PrRd** או **PrWr** בהתאם לצורך.

בפונקציות הנ"ל בודקים לפי פרוטוקול MESI מה אנחנו רוצים לעשות עם הבקשה הזאת. במידה ואין צורך להשתמש בBUS כלל, שמים DONE במבנה של הבקשה, אחרת, שמים בקשה של BUS\_REQ.

עבור הבקשות מסוג BUS\_REQ יש מספר מערכים שעוזרים לנו לבצע את הארביטרציה של הבקשות. כאשר יש בקשה חדשה, מדליקים 1 במערך הבקשות core\_has\_request ושמים מצביע לבקשה ב pr\_requests.

הפונקציה העקרית של הBUS היא **bus\_step** והיא זאת שהקוד הראשי קורא לה בכל מחזור שעון: ראשית, בודקים האם bus\_cmd הוא 0, אם כן זה אומר שאין בקשה נוכחית ולכן צריך לבדוק האם יש בקשות אחרות לשרת. את זאת עושים באמצעות הפונקציות:

**choose\_core**: בודקת לפי סדר הround robin האם יש בקשה חדשה לשרת, אם לא, נצא מהפונקציה, אם כן, מחזירה את הליבה הרלוונטית.

**core\_used\_bus**: מעדכנת את המערכים של ה round robin.

לאחר מכן קוראים לפונקציה **place\_request\_on\_bus** שתפקידה הוא לתרגם את ה BUS\_REQ לקווים של הBUS עצמו.

אחר כך קוראים לפונקציה **bus\_logic\_before\_snooping** שתפקידה הוא לבדוק האם סיימנו את הספירה לאחור עבור קריאה מהזכרון הראשי וגם לשים את מה שרוצים לעשות לו flush על הקווים.

לאחר מכן קוראים לפונקציה **print\_bus\_trace\_line** ששמה בקובץ הפלט את תוכן הBUS.

אחר כך עושים snoop לכל ליבה באמצעות **core\_i\_snoop** ולזכרון הראשי באמצעות **main\_mem\_snoop**.

אחר כך קוראים ל **bus\_logic\_after\_snooping**שתפקידה הוא לבדוק האם סיימנו לעשות FLUSH ולעדכן כל מיני counters.

לבסוף קוראים ל **check\_if\_req\_fulfilled**שבודקת האם סיימנו לעשות את הבקשה הנוכחית ואם כן מעדכנת את מה שצריך.